Universidad Tecnológica Centroamericana
Facultad de Ingeniería

CC414 - Sistemas Inteligentes
Docente: Kenny Dávila, PhD

Tarea #3 (4% Puntos Oro)

Para completar esta tarea es requerido usar **Python 3** y la librería **Scikit-Learn**. También se recomienda el uso de las librerías **Numpy** para manejar los datos y Matplotlib para la creación de plots con los datos resultantes.

Objetivo

El objetivo de este proyecto es evaluar el uso de **árboles de decisión** y **bosques aleatorios** utilizando la librería Scikit-Learn.

Contexto

En la actualidad existe una gran variedad de cadenas de comidas rápidas. A pesar de que muchas cadenas de estas cadenas ofrecen un menú similar, las mismas logran coexistir gracias a una serie de diferencias en múltiples aspectos como ser sabor, precio y muchos otros. Estas diferencias hacen que algunos clientes tengan una opinión positiva o negativa de sus comidas. En este caso en particular, se consideran 4 cadenas de restaurantes de hamburguesas. Se desea crear un sistema de recomendaciones de restaurante de hamburguesas en base a preferencias del usuario. En esta tarea usted tendrá la oportunidad de generar sus propias hipótesis en función de datos simulados provenientes de una distribución oculta.

Problema

Dado un conjunto de datos de entrenamiento (training dataset), usted deberá utilizar los algoritmos de aprendizaje de árboles de decisión y de bosques aleatorios para crear múltiples predictores de restaurante de hamburguesas preferidos. En este caso se consideran 4 cadenas de restaurantes: “Mendys”, “WAC Ronalds”, “Burger Queen” y “Rigos”. Dado un vector nuevo que representa las preferencias de un cliente específico, usted deberá usar el algoritmo entrenado, ya sea un árbol de decisión o un bosque aleatorio, para intentar predecir qué restaurante será el preferido por dicha persona.

Atributos

La selección correcta de atributos es importante al momento de generar un algoritmo que sea capaz de predecir una clase a partir de los datos de entrada. Para esta tarea, se tiene disponible de la siguiente información sobre cada persona:

1. **Jugosas**. Indica si la persona prefiere hamburguesas hechas con carne muy jugosa.
2. **Carne Fresca**. Indica si la persona prefiere hamburguesas hechas con carne fresca.
3. **Opciones Vegetarianas.** Indica si la persona prefiere opciones vegetarianas.

4. **Opciones de Quesos**. Se refiere a si la persona gusta probar hamburguesas con diferentes tipos de quesos.
5. **Malteadas.** Indica si a la persona le gustan las malteadas.
6. **Con Huevo.** Indica si a la persona le gustan las hamburguesas con huevo.
7. **Buenas Papas**. Indica si la persona es exigente en cuanto al sabor de las papas fritas que se sirven junto con las hamburguesas.
8. **Juguetes.** Se refiere a si la persona a menudo compra combos que incluyan un juguete.
9. **Con Hongos**. Se refiere a si a esta persona le gusta comer hamburguesas con champiñones.
10. **Con Pepinillos.** Indica si a la persona le gustan las hamburguesas con pepinillos.
11. **Combos Familiares**. Se utiliza para identificar si la persona a menudo compra combos para toda la familia.
12. **Rapidez**. Indica si la persona prefiere que su comida sea entregada lo más rápido posible.
13. **De Pollo**. Marca si la persona a menudo come sándwiches de pollo.
14. **De Pescado**. Indica si la persona a menudo come sándwiches de pescado.
15. **Mas Salsa**. Se refiere a si la persona prefiere hamburguesas con mucha salsa.

Nótese que todos los atributos descritos anteriormente son binarios y solamente pueden tener un valor de "si" o "no". Tanto el dataset de entrenamiento como el dataset de pruebas son provistos en formato de valores separados por comas (CSV). Estos archivos son fáciles de manipular de manera nativa con Python, pero también existen múltiples librerías que facilitan la carga y manipulación de dichos archivos (e.g., la librería Pandas).

Experimentos

El objetivo de este este ejercicio es utilizar las implementaciones de Descision Tree Classifier y Random Forest Classifier provistas por la librería **Scikit-Learn**. Debe utilizar los datos de clasificación en 4 categorías ("Mendys", "WAC Ronalds", "Burger Queen" y "Rigos") y evaluar el rendimiento de cada clasificador sobre dichos datos. Nótese que será necesario que convierta los atributos binarios en números (0 y 1s) antes de poder los clasificadores sobre estos datos.

1) **Decision Trees:** Se le pide usar los diferentes datos de entrenamiento ("training_data_small.csv", "training_data_medium.csv", "training_data_large.csv", "training_data_very_large.csv") para entrenar diferentes modelos del árbol de decisión usando diferentes combinaciones de los criterios disponibles ("Gini" y "entropy") y max_depth (2, 4, 8, y None). Se espera que se explore al menos **un total de 32 configuraciones** de manera sistemática (4 datasets de entrenamiento x 2 criterios x 4 profundidades máximas). Otros parámetros, como "splitter", "max_features", "min_impurity_decrease" y "max_leaf_nodes" entre otros, deben dejarse en sus valores por defecto.

   Deberá utilizar los datos de validación ("validation_data.csv") para calcular y reportar las siguientes métricas de evaluación para cada configuración de árbol: Accuracy total; valores promedio de recall, precisión y F1-score para todas las clases; y también tiempo total de predicción. Adicionalmente, se le pide calcular el accuracy total en los datos de

entrenamiento que se utilizaron para entrenar cada configuración, y de esta forma poder analizar si hay overfitting. Debe usar una sola tabla para reportar dichos resultados. Debe utilizar **al menos 4 cifras significativas** por cada valor provisto.

Tabla de resultados de ejemplo (Arboles de Decisión)

Train Dataset	Criterion	Depth	Train Acc.	Val. Acc.	Val. Avg Rec.	Val. Avg Prec.	Val. Avg F1	Time Train	Time Val.
Small	Gini	2	0.XXXX%	0.XXXX%	0.XXXX%	0.XXXX%	0.XXXX%	T s	T s
…	….	…	…	…	…	…	…	…	…
Very Large	Entropy	None	0.XXXX%	0.XXXX%	0.XXXX%	0.XXXX%	0.XXXX%	T s	T s

2) **Random Forests:** Debe utilizar los mismos datos de entrenamiento (4 conjuntos diferentes) para entrenar diferentes modelos del random forest usando diferentes combinaciones de los valores de n (5, 10, 50, 100) y max_depth (2, 4, y 6). Se espera que se explore al menos **un total de 48 configuraciones** de manera sistemática (4 datasets de entrenamiento x 4 valores n x 3 profundidades máximas). Otros parámetros deben dejarse en sus valores por defecto. Deberá utilizar los datos de validación ("validation_data.csv") para calcular y reportar las mismas métricas solicitadas para los árboles de decisión. Debe usar una sola tabla para reportar dichos resultados. Debe utilizar **al menos 4 cifras significativas** por cada valor provisto.

Tabla de resultados de ejemplo (Bosques Aleatorios)

Train Dataset	N trees	Depth	Train Acc.	Val. Acc.	Val. Avg Rec.	Val. Avg Prec.	Val. Avg F1	Time Train	Time Val
Small	5	2	0.XXXX%	0.XXXX%	0.XXXX%	0.XXXX%	0.XXXX%	T s	T s
…	….	…	…	…	…	…	…	…	…
Very Large	100	6	0.XXXX%	0.XXXX%	0.XXXX%	0.XXXX%	0.XXXX%	T s	T s

Requisitos

1. Implementación de un script basado en Python que recibe como entrada el nombre del archivo de datos de entrenamiento y otros parámetros del árbol y entrene un árbol de decisión que deberá ser guardado en un archivo de salida.

2. Implementación de un script basado en Python que recibe como entrada el nombre del archivo de datos de entrenamiento y otros parámetros del bosque aleatorio y entrene un bosque aleatorio que deberá ser guardado en un archivo de salida.

3. Implementación de un script basado en Python que recibe como entrada el nombre del archivo donde se guardó el clasificador (ya sea árbol de decisión o bosque aleatorio), y el nombre del archivo con los datos de prueba, y que produzca como salida una matriz de confusión y las métricas correspondientes sobre los datos de prueba. Note que gracias al "duck typing" de Python, no necesitara hacer nada especial para que el mismo código funcione bien con un tipo de clasificador o con el otro.

4. Un reporte que detalle lo siguiente:
    a. **Introducción**. Debe incluir una breve motivación del problema en sus propias palabras
    b. **Implementación.** Descripción de su implementación incluyendo comandos específicos para ejecutar su código fuente y librerías específicas de Python que son requeridas por su programa.
    c. **Resultados.** Debe proveer las tablas con todos los resultados que se obtuvieron de sus experimentos con diferentes datos de entrenamiento y configuraciones de cada clasificador.
    d. **Análisis.** Entre muchos detalles importantes que deberían discutirse, debe como mínimo analizar lo siguiente:
        i. ¿Cuáles fueron los 3 atributos más importantes según el árbol de decisión entrenado? y un breve análisis en sus propias palabras de porque consideran que estos atributos fueron seleccionados por el algoritmo.
        ii. ¿Cuáles fueron los 3 atributos menos importantes? y ¿por qué considera que no fueron tan útiles durante la clasificación?
        iii. Análisis del efecto de la profundidad del árbol en cuanto a rendimiento y overfitting.
        iv. Análisis del efecto del tamaño del dataset en cuanto a rendimiento y overfitting.
        v. Análisis del efecto del criterio utilizado para determinar la relevancia de los atributos en cuanto a rendimiento y overfitting.
        vi. Sorpresas. Detalle si los resultados anteriores fueron consistentes o no con sus propias expectativas sobre el tema.
    e. **Dificultades encontradas.** Describa en sus propias palabras cuales fueron los retos más difíciles al momento de implementar el algoritmo.
    f. **Planteamiento**. Debe presentar un problema distinto donde usted aplicaría árboles de decisión o bosques aleatorios, con atributos que podría colectar sobre ese problema y una corta justificación (Mínimo 300-Máximo 500 palabras).
    g. Conclusiones

Otras políticas

1. Esta tarea deberá trabajarse y entregarse **individual** o en **parejas.**
2. La entrega será **un solo archivo comprimido (.zip o .rar)**. Dentro de dicho archivo debe contener la guía completada **en formato PDF**. También debe los scripts de Python que se usaron para contestar cada punto.
3. Favor **NO incluir entornos virtuales ni datasets dentro** de su archivo Zip. La entrega final no debería pasar de 1MB.
4. No se permite el uso de repositorios públicos.
5. Si se hace en parejas, ambas personas deben subir el mismo archivo.
6. El **plagio** será penalizado de manera severa.
7. Si se hace en parejas, y hay plagio, **ambos miembros serán considerados igualmente responsables por el plagio**. Sin excepciones.
8. Los estudiantes que entreguen una tarea 100% original recibirán una nota parcial a pesar de errores existentes. En cambio, los estudiantes que presenten tareas que contenga material plagiado recibirán 0% automáticamente independientemente de la calidad, y serán reportados al comité de ética de Unitec.
9. Tareas entregadas después de la fecha indicada solamente podrán recibir la mitad de la calificación final. Por esta razón, es posible que **un trabajo incompleto pero entregado a tiempo termine recibiendo mejor calificación que uno completo entregado un minuto tarde**.